ECEC 412/621: Computer Architecture
Solution Key for Exam II (Spring 2015)

June 4, 2016

1. The following C program is run with no optimizations on a processor with a cache that has eight-word (32 byte) blocks and holds 256 bytes of data:

```
int i, j, c, stride, array[512];
...
for(i = 0; i < 10000; i++)
    for(j = 0; j < 512; j = j + stride)
        c = array[j] + 10;
```

Assume that the only cache activity is generated by references to the array and that integers are words. What is expected miss rate when the cache is direct mapped and stride = 256? How about if stride = 255? Would either of these change if the cache were two-way set associative?

The direct mapped cache has eight blocks where the size of the block is eight words or 32 bytes. When `stride = 256`, the access pattern for the elements of `array` is as follows:

```
array[0], array[256], array[0], array[256], ...
```

The address corresponding to `array[0]` and `array[256]` both map to cache block 0.[1] So, these competing accesses will result in conflict misses in the cache. The miss rate is 100%.

When `stride = 255`, the access pattern for the elements of `array` is as follows:

```
array[0], array[255], array[510], array[0], array[255],
array[510], ...
```

The address for `array[0]` maps to cache block 0, whereas the addresses for `array[255]` and `array[510]` both map to cache block 7. So the miss rate is 66.7%. If we switch to a two-way set associative cache, after the initial compulsory misses, the miss rate will be 0%. When `stride = 256`, the addresses corresponding to `array[0]` and `array[256]` will both map to set 0, and they can be placed in the two cache blocks residing in set 0. Similarly, when `stride = 255` the addresses for `array[255]` and `array[510]` both map to cache set 3, and can be placed in the two blocks belonging to this set.

[1] The address for `array[256]` resides in memory block $\lfloor 256/8 \rfloor = 32$, and memory block 32 maps to cache block $32 \bmod 8 = 0$.

2. Show the design of a four-way set associative write-back cache of size 256 KB having 32-byte blocks. Also, show how a 32-bit main memory address is decomposed to access the cache. Assume that memory is word addressable. What is the final size, including tags and status bits, in KB of the cache structure?

The 32-bit memory address is partitioned as follows:

1. *Offset bits*: We choose three bits, 4–2, to locate the word within a cache block. Note that each cache bock contains eight word. Bits 1 and 0 can be used to locate a specific byte within the word, but we have assumed that memory is word addressable. So, three offset bits (4–2) are needed.
2. *Index bits*: Using the formula,

$$2^{\text{index}} = \frac{256 \times 1024}{32 \times 4} = 2048 \text{ sets},$$

   we get 11 index bits. These are bits 15–5 in the 32-bit memory address.
3. *Tag bits*: The remaining 16 bits (31–16) are used as the tag bits.

The final size of the cache structure includes the overhead due to the tag bits, the dirty bit (since we are dealing with a write-back cache) and the valid bit: $(32 \times 8 + 16 + 1 + 1) \times 4 \times 2048$ bits or 274 KB.

3. Consider a two-level instruction cache hierarchy comprising direct-mapped level one and level two caches, L1 and L2, respectively. Suppose that in 2000 memory references there are 50 misses in the L1 cache and 3 misses in the L2 cache. Also, assume the following:

- Hit time in the L1 cache is 1 processor cycle.
- Hit time in the L2 cache is 5 processor cycles.
- Miss penalty incurred in the L2 cache is 30 processor cycles.

(a) What is the local miss rate for the L2 cache?

Miss $\text{rate}_{\text{L2}} = 3/50 = 0.06$.

(b) What is the average instruction access time for the above cache hierarchy?

Hit $\text{time}_{\text{L1}} = 1$ cycle.

Miss $\text{rate}_{\text{L1}} = 50/2000 = 0.025$.

Hit $\text{time}_{\text{L2}} = 5$ cycles.

Miss $\text{rate}_{\text{L2}} = 3/50 = 0.06$.

$$\begin{aligned}\text{AMAT} &= \text{Hit time}_{\text{L1}} + \text{Miss rate}_{\text{L1}} \times (\text{Hit time}_{\text{L2}} + \text{Miss rate}_{\text{L2}} \times \text{Miss penalty}_{\text{L2}}) \\ &= 1 + 0.025 \times (5 + 0.06 \times 30) \\ &\approx 1.17\end{aligned}$$

4. In practice, the set-associativity index of a cache is always a power of 2 (1-way, 2-way, 4-way, 8-way, etc.). Design a functional 3-way set-associative cache (your cache should work correctly!) that can store up to 33 KB of data with a block size of 32 bytes. Assume that main memory is byte addressable and that a 32-bit main memory address is used to access the cache. From this design experience, indicate why you believe 3-way set-associative caches are not built in practice.

Using the formula,

$$2^{\text{index}} = \frac{33 \times 1024}{32 \times 3} = 352 \text{ sets},$$

we use 8 index bits which will allow us to index 256 sets. We cannot use 9 index bits (for accessing 512 sets) since if we did that, there will be main memory addresses that will map to non-existent locations in the cache; that is to sets 352 through 511 which do not exist in hardware. Since we only use 256 sets, $352 - 256 = 96$ sets or $(96 \times 3 \times 32)/1024 = 9$ KB of the cache capacity goes unused.

Note that some of you used 9 bits to index into the cache while ensuring that memory addresses that map to sets 352 through 511 are remapped to be within the range 0 through 351. I awarded full points for such a design. Note however that the remapping hardware (say a modulo 352 circuit) will significantly slow down the cache making it all but unusable.